CppUnit Tutorial

Prepared by Zaid Towfic on 1/3/04

Downloading the package:

The first step is to download CppUnit. Download it from here:
http://sourceforge.net/project/showfiles.php?group_id=11795
I am using version 1.8.0.

After downloading the package to the desktop, for example, extract the archive.
So a sample path would be: C:\Documents and Settings\gtowfic\Desktop\cppunit-1.8.0

Compiling the package:

1. Go the src directory in the root directory of the package, so you would be here:
C:\Documents and Settings\gtowfic\Desktop\cppunit-1.8.0\src
2. Open the CppUnitLibraries.dsw file and convert all files to version 7.0 when prompted.
3. Build the package by going to Build->Build Solution or hitting Shift+F8. You will be prompted to save a solution file, so just save it in the same directory. Ignore any error messages or linker messages.
4. Go to the “msvc6” directory then to the “testrunner” directory, so you would be here:
C:\Documents and Settings\gtowfic\Desktop\cppunit-1.8.0\src\msvc6\testrunner
5. Open the TestRunner.dsw file and convert it to version 7.0 when prompted.
6. Build the package by going to Build->Build Solution or hitting Shift+F8. You will be prompted to save a solution file, so just save it in the same directory.
You are done compiling the package!

Running some tests:

1. Go back to the root directory and then to the directory “lib”. Copy the file "testrunnerd.dll" and paste it in the directory C:\Documents and Settings\gtowfic\Desktop\cppunit-1.8.0\examples\msvc6\CppUnitTestApp\Debug (create the directory). Go back up one directory, so you should be here: C:\Documents and Settings\gtowfic\Desktop\cppunit-1.8.0\examples\msvc6\CppUnitTestApp.dsp
2. Open CppUnitTestApp.dsp and convert it to 7.0 when prompted.
3. Again, build the package by going to Build->Build Solution or hitting Shift+F8. You will be prompted to save a solution file, so just save it in the same directory.
4. After a successful compilation, run the program by going to Debug->Start or hitting F5.

Adding a test case:

1. Now we will need to create a test case. This test case will use two classes: Student and Course.
Here's the source code for the classes (from http://tramp.globalse.org/doc/presentations/testingTutorial.pdf):

ALL THESE FILES HAVE TO BE IN THIS DIRECTORY:
C:\Documents and Settings\gtowfic\Desktop\cppunit-1.8.0\examples\cppunittest

```
//************************Course.h************************
#ifndef Course_h
#define Course_h
#include <string>
class Course
{
public:
// Default constructor
Course();
// Constructor
Course(std::string nm, int gr);
// Method to get the name of the course
std::string getCourseName();
// method to get the grade of the course
int getCourseGrade();
private:
std::string course_name; // name of this course
int grade; // grade of this course
};
#endif
```

```
//************************Course.cpp************************
#include "Course.h"
// Default constructor
Course::Course()
{
course_name = "";
grade = -1;
}
// Constructor
Course::Course(std::string nm, int gr):course_name(nm)
{
grade = gr;
}
// Method to get the name of the course
```

```
std::string Course::getCourseName()
{
return course_name;
}
// Method to get the grade of the course
int Course::getCourseGrade()
{
return grade;
}


//************************Student.h************************
#ifndef Student_h
#define Student_h
#include <iostream>
#include <string>
#include "Course.h"
const int MAXNUM = 20; // Maximum number of courses allowed per student
class Student
{
public :
// Constructor
Student(std::string nm, std::string no);
// Method to return student's name
std::string getStuName();
// Method to return student's number
std::string getStuNumber();
// Method to assign a grade to a course
void assignGrade(std::string co, int gr);
// Method to return the grade of a course
int getGrade(std::string co);
private:
std::string name; // name of the student
std::string number; // the student's number
Course course_grades[MAXNUM]; // courses taken by student
int no_of_courses; // the current number of courses taken
};
#endif

//************************Student.cpp************************
#include "Student.h"
// Constructor
Student::Student(std::string nm, std::string no):name(nm), number(no)
{
no_of_courses = 0;
}
```

```
// Method to return student's name
std::string Student::getStuName()
{
return name;
}
// Method to return student's number
std::string Student::getStuNumber()
{
return number;
}
// Method to assign a grade to course
void Student::assignGrade(std::string co, int gr)
{ // Check whether the maximum number of courses have been taken
if (no_of_courses == MAXNUM)
{
std::cout << "You have exceeded the maximum number of courses !\n";
return;
}
// Create a new course
Course c(co, gr);
course_grades[no_of_courses++] = c;
}
// Method to return the grade of a course
int Student::getGrade(std::string co)
{
int i = 0;
while (i < no_of_courses)
{
//check if course name the same as co
if (course_grades[i].getCourseName() == co)
return (course_grades[i].getCourseGrade());
i++;
}
return(-1);
}
```

2. Ok, now we will need to write the test case:
   The test case will have two files (since it is a class): TestStudent.h and TestStudent.cpp
   Here's a template for a test case:

ALL THESE FILES HAVE TO BE IN THIS DIRECTORY:
C:\Documents and Settings\gtowfic\Desktop\cppunit-1.8.0\examples\cppunittest

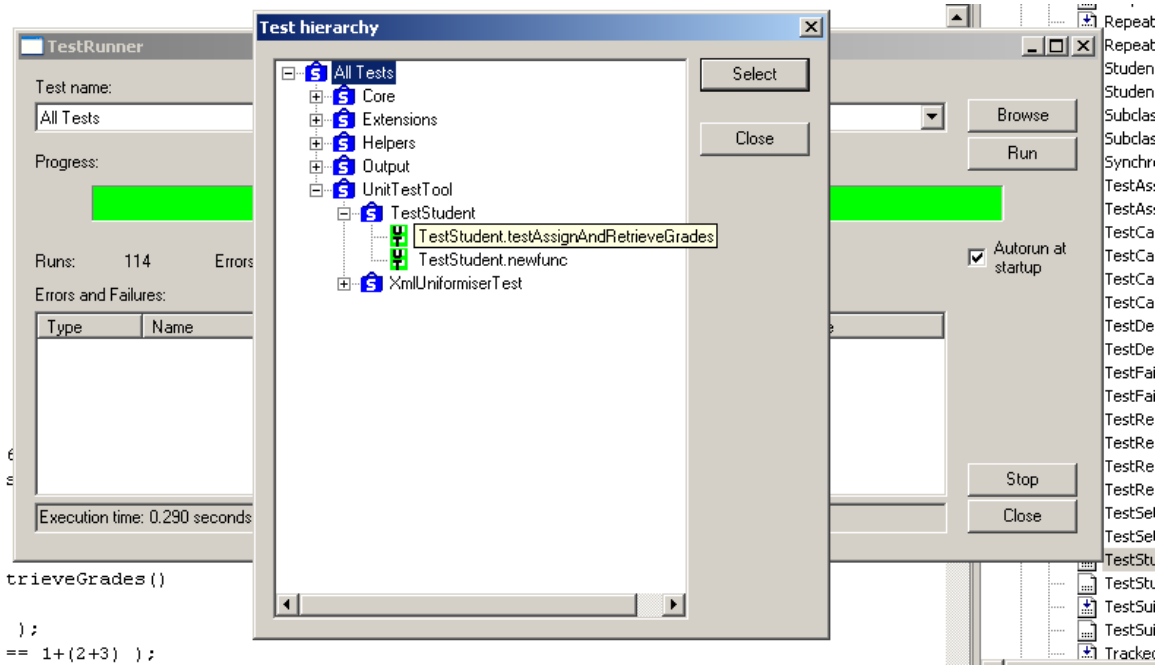
```
//**********************TestStudent.h**********************
#ifndef TESTSTUDENT_H
#define TESTSTUDENT_H
```

```
#include <cppunit/extensions/HelperMacros.h>


class TestStudent : public CppUnit::TestFixture
{

  CPPUNIT_TEST_SUITE( TestStudent );
  CPPUNIT_TEST( testAssignAndRetrieveGrades );
  CPPUNIT_TEST( newfunc ); // NEW FUNCTION ============================
  //Add test functions here-----------------------------------------
  CPPUNIT_TEST_SUITE_END();

public:
  /*! Constructs a TestStudent object.
   */
  TestStudent();

  /// Destructor.
  virtual ~TestStudent();

  void setUp();
  void tearDown();
  void testAssignAndRetrieveGrades();
  void newfunc(); //================================================
private:

  /// Prevents the use of the copy operator.
  void operator =( const TestStudent &copy );

private:
};

#endif  // TESTSTUDENT_H




//************************TestStudent.cpp************************
#include <cppunit/NotEqualException.h>
#include <stdlib.h>
#include "UnitTestToolSuite.h"
#include "TestStudent.h"
#include "Student.h"

CPPUNIT_TEST_SUITE_NAMED_REGISTRATION( TestStudent,
CppUnitTest::unitTestToolSuiteName() );


TestStudent::TestStudent()
{
}
```

```
TestStudent::~TestStudent()
{
}


void
TestStudent::setUp()
{
}


void
TestStudent::tearDown()
{
}

void
TestStudent::newfunc()          // NEW FUNCTION=========================
{                               // =====================================
      Student stu("Jimmy", "946302B"); //==============================
    CPPUNIT_ASSERT ( -1 ==  stu.getGrade("cs21002"));//================
}

void
TestStudent::testAssignAndRetrieveGrades()
{
    CPPUNIT_ASSERT( 10 == 10 );
    CPPUNIT_ASSERT( (1+2)+3 == 1+(2+3) );
      Student stu("Jimmy", "946302B");
// Assign a few grades to this student
stu.assignGrade("cs2102", 60);
stu.assignGrade("cs2103", 70);
stu.assignGrade("cs3214s", 80);
// Verify that the assignment is correct - Note 6
CPPUNIT_ASSERT( 60 == stu.getGrade("cs2102"));
CPPUNIT_ASSERT(70 == stu.getGrade("cs2103"));
// Attempt to retrieve a course that does not exist
CPPUNIT_ASSERT(-1 == stu.getGrade("cs21002"));

}
```

If you want to add more functions to test, then they should be placed in the header file. The header file also tells the GUI where to place the functions (line 13).

3. Ok, now we will add the files to the project. Go to the CppUnitTestApp (in your Visual Studio .NET) and right click on the “Tests” folder (in your Solution Explorer) and select Add->”Add existing Item” then select the files from your “C:\Documents and Settings\gtowfic\Desktop\cppunit-1.8.0\examples\cppunittest” (The files are: Course.h, Course.cpp, Student.h, Student.cpp, TestStudent.h, TestStudent.cpp)

4. Now we will need to disabled the precomiled headers. To do this, right click on your project in your Solution Explorer (CppUnitTestApp) then properties. In the C\C++ category, select the “Precompiled Headers” sub-directory and under the “Create/Use precompiled headers” select “Not Using Precompiled Headers”.

5. We should be done. Try to build and run your application. Your “TestStudent” test should be listed under “UnitTestTool” when you select Browse in the application: